Robot Raconteur® Security using TLS and Certificates

Version 0.8 Beta

http://robotraconteur.com

Dr. John Wason

Wason Technology, LLC

PO Box 669

Tuxedo, NY 10987

wason@wasontech.com

http://wasontech.com

May 3, 2016

Contents

1 Introduction

Network security is a growing problem for industrial automation and distributed internet systems (Internet of Things). Robot Raconteur® provides security through a combination of Transport Layer Security (TLS) $\geq$ 1.0 and X509 certificates issued by Wason Technology, LLC in partnership with Symantec. TLS provides strong security using RSA encryption combined with a symmetric encryption algorithm, most commonly AES. AES provides military grade encryption, and because most modern processors contain hardware acceleration, performance is very high. Certificates are used to validate the identity of nodes communicating using 4096-bit RSA keys. The use of TLS over the Robot Raconteur `TcpTransport` is transparent to the user requiring only a change in the URL. Using TLS requires installation of an identity X509 certificate on the service node and optionally the client node if certificates are used for authentication.

2 Node Identity Certificates

Identity certificates are an important part of TLS and provide a way to guarantee that the connected peer is not an imposter. Certificates use what is called “Asymmetric Encryption”. “Asymmetric Encryption” keys have two parts: a public key that is globally available, and a private key that is kept secret and known only to the assigned node. The public key is encased in a “Certificate” that is “signed” by an “authority” that guarantees the identity of the certificate. The private key is installed on the server/device that corresponds to the identity. When a client connects to the server/device, it compares the certificate of the server/device to the “root certificate” distributed by the authority. This root certificate is published and is known by all devices that need to authenticate certificates.

Certificates are issued by Wason Technology, LLC to guarantee the Node Id identity of the assigned node. A NodeID is a 128 bit UUID that can be represented as a string of hexadecimal

numbers. An example UUID is:

{8afda3b2-f70b-4b41-8086-71b0b3415e86}

The assigned certificate guarantees that the Node ID is unique and that only one device will have a certificate assigned to this NodeID.

3 Certificate Distribution

Certificates for Robot Raconteur nodes are sold by Wason Technology, LLC and can be purchased on the Robot Raconteur website at http://robotraconteur.com/nodeca/purchase. Once purchased, "tokens" are added to your account which can then be redeemed for node certificates. Tokens can be viewed at http://robotraconteur.com/nodeca as well as a list of certificates that have been issued using your purchased tokens. Certificates are issued using the *Robot Raconteur Certificate Utility* program that can be downloaded at http://robotraconteur.com/download. The utility can either be run as a graphical program or as a command line program.

3.1 Robot Raconteur Certificate Utility usage

The graphical tool can be downloaded from http://robotraconteur.com/download. The ZIP file provided contains a universal program that uses .NET on Windows and Mono on Linux or Mac OSX.

Windows

.NET 4.5 is required. Simply unzip the files and run the gui or command line tool by double clicking on the executable or using the command prompt.

Linux

Mono 4.0 or greater is required to run the certificate utilities, but most Debian based distributions are still running version 3.18 of Mono. To install the newest version of Mono, follow the instructions on the Mono website:

http://www.mono-project.com/docs/getting-started/install/linux/

Warning: Do not follow these instructions on the Raspberry Pi A, Raspberry Pi B, Raspberry Pi B+, or Raspberry Pi Zero. The Mono installation is intended for ARMv7, and the older Raspberry Pi devices are ARMv6. The Raspberry Pi 2 and Raspberry Pi 3 are ARMv7 and will work correctly.

Next, install `mono-complete`:

```
sudo apt-get install mono-complete
```

Unzip the downloaded certificate utility file, and run the utilities.

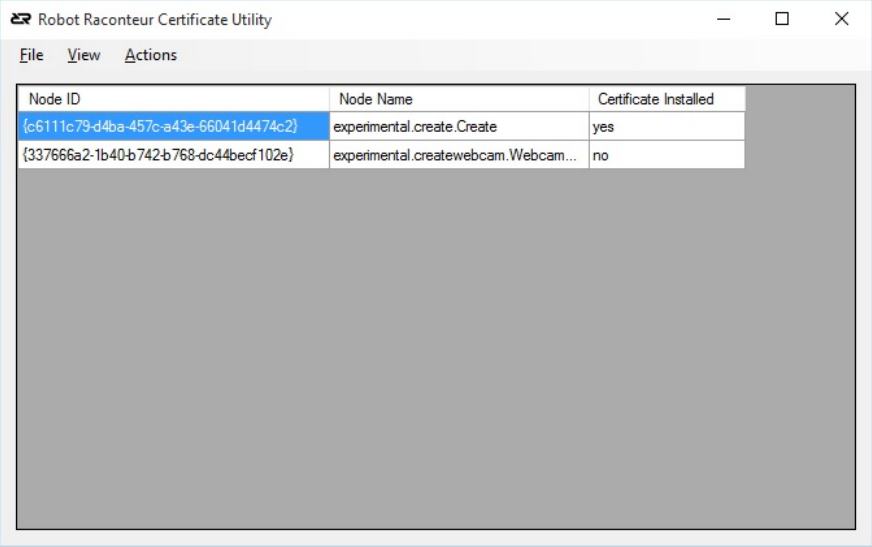

Figure 1: The certificate utility base window

```
mono RobotRaconteurCertificateUtility
```

or

```
mono RobotRaconteurCertificateUtility-gui
```

Mac OSX

The certificate utility requires Mac Mono to be installed. Unfortunately this is quite large. Future versions will not require this additional installation.

http://www.mono-project.com/download/.

Once Mono is installed, unzip the downloaded certificate utility file, and run the utilities.

```
mono RobotRaconteurCertificateUtility
```

or

```
mono RobotRaconteurCertificateUtility-gui
```

3.2 Certificate Installation using the Graphical User Interface Utility

The “Robot Raconteur Certificate Utility GUI” is used to manage local NodeIds, NodeNames, and node certificates. The base window shows the current nodes that were detected on the computer, and whether they have a certificate installed. See Figure 1.

Right clicking on a node brings up a context window. See Figure 2. In this case, we right click on

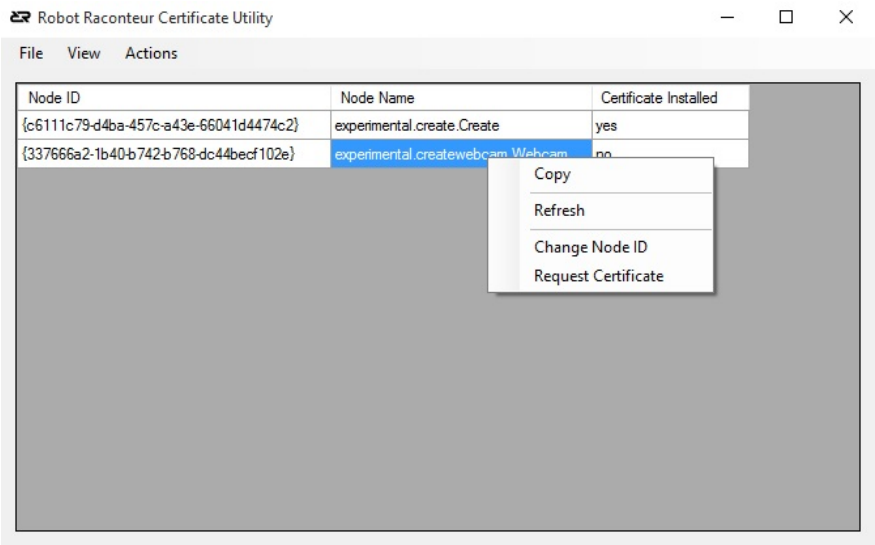

Figure 2: Context menu

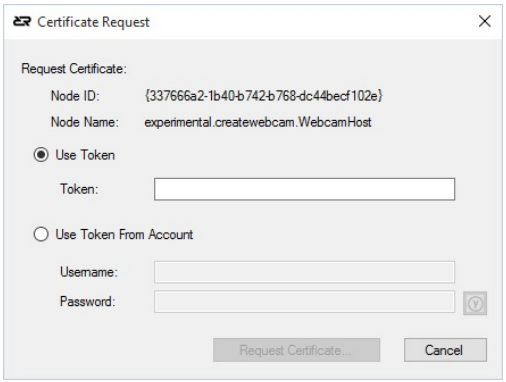

Figure 3: The certificate request window

"experimental.createwebcam.WebcamHost" since it does not have a certificate. Click on "Request Certificate" This brings up the certificate request window. See Figure 3. In this window, you can either enter a purchased token, or enter your username and password for robotraconteur.com to use the next available token. You must purchase tokens from http://robotraconteur.com/nodeca before you can request a certificate. Click "Request Certificate". This operation may take some time. Stay connected to the Internet until the process is complete.

If the operation fails, contact us at http://robotraconteur.com/contact-us

Sometimes it may be necessary to change the NodeID assigned to a NodeName. This can be done using the "Change Node ID" command in the context menu seen in Figure 2. Click "Generate" to create a new unique ID, or enter one manually.

3.3 Certificate Installation using the Graphical Command Line Utility

The command line utility provides the ability to issue certificates on a limited device or in any situation where it is more convenient. Note that on Linux and Mac OSX all commands must be preceded with "mono", ie `mono RobotRaconteurCertificateUtility`.

To list the nodes and there associated NodeName/NodeId that are currently in use on the local device, run:

```
RobotRaconteurCertificateUtility --list-nodes
```

To install a certificate to the local device, use this command. Replace the token and node id with your desired values. (Note that all the following should be on one line)

```
RobotRaconteurCertificateUtility --request-certificate
   --nodeid=bf38ec84-5198-4797-b6e1-dd38df20ac79
   --token=RJT8Z64EZ1NSL26NQJCU75X7JOZPSKSZ
```

Sometimes it is necessary to install a certificate on a different device from the device that is requesting the certificate. The command line utility can request a certificate and return a PKCS#12 file that contains the certificate chain and private key. The following command will return a file named "{bf38ec84-5198-4797-b6e1-dd38df20ac79}.p12" in the working directory.

```
RobotRaconteurCertificateUtility --request-certificate
   --nodeid={bf38ec84-5198-4797-b6e1-dd38df20ac79}
   --token=RJT8Z64EZ1NSL26NQJCU75X7JOZPSKSZ    --tofile
```

To install this file on a Windows device, double click on the file and import it into your local certificate store using the wizard with default options. For Linux and Mac OSX, copy the file to the configuration directory:

```
$HOME/.config/RobotRaconteur/certificates
```

The example commands to run at the command prompt to install the certificate:

```
mkdir -p $HOME/.config/RobotRaconteur/certificates
cp {bf38ec84-5198-4797-b6e1-dd38df20ac79}.p12 $HOME/.config/RobotRaconteur/certificates/
```

4 Securing Nodes

Several strategies can be used to help secure Robot Raconteur nodes against attack. Example **??** shows an example secure service, and Example **??** shows an example client based on the following strategies:

1. Require TLS

All TCP connections should require TLS security. Activating TLS requires three steps: assigning the node id, loading the certificate, and settings the transport to require TLS. The easiest way to assigning the node id is to use `LocalTransport.StartServerAsNodeName` which will pull the node id from local settings. A certificate must be issued using the certificate utility before it can be loaded. The following are the relevant lines from Example **??**:

```
localtransport.StartServerAsNodeName('example.securenode')
tcptransport.LoadTlsNodeCertificate()
tcptransport.RequireTls=True
```

2. Require Password and Certificate authentication

Robot Raconteur nodes can be secured with an authenticator that requires a username and typically a password. This must be done in all cases to protect the node. For further security, the authenticator can also check if the client has supplied a certificate. Client nodes load the same certificates as server nodes and identify the client's node id. Example **??** contains an implementation of the class `CertificateAuthenticator` which will check both passwords and the certificate. See Example **??** and Example **??** for an example of client certificate based authentication. The most import line from the example is:

```
clientnodeid=tcptransport.GetSecurePeerIdentity(RR.ServerEndpoint.GetCurrentEndpoint())
```

This line is in the authenticator and will retrieve the node id of the client that is verified by the client certificate.

3. Specify secure connection URL

Connection URLs identity the transport mechanism, the target node endpoint, the service to connect, and optionally the NodeID/NodeName to connect. While it is possible to omit the NodeID/NodeName in many situations, it is best practice to always specify the NodeID when connecting to a remote node, and a TLS should URL should always contain the NodeID to prevent impersonation. If the NodeID is not specified, the identity of the service is not verified. For example, this is a good TLS connection URL:

```
rrs+tcp://101.2.2.2/?nodeid=6f6706c9-91cc-d448-ae8c-c5a2acac198c&service=Create
```

The NodeName is not guaranteed to be unique, and is available for convenience. It should not be used when creating TLS connections.

4. Limit the message size and connection count

Denial-of-Service (DOS) and Distributed Denial-of-Service (DDOS) are among the most common types of network attacks. These attacks work by generating high numbers of requests that can quickly overwhelm devices. Because most Robot Raconteur devices have limited resources devoted to communication and are only designed to service a few clients they can rapidly become overwhelmed. The best defense against this type of attack is to limit the maximum message size and the maximum number of concurrent connections. The maximum message size is by default 12 MB which is much larger than is necessary for most devices. The maximum connection count is unlimited by default and should be set to 8 for a device that only expects to service one client at a time. (Note that the autodiscovery will also create short lived connections so a minimum of 8 should be specified.) The following are the relevant lines from Example **??**:

```
tcptransport.MaxMessageSize=12*1024
tcptransport.MaxConnectionSize=8
```

5. Secure WebSocket connections

By default, Robot Raconteur will accept WebSocket connections. WebSockets are an HTTP protocol that allow for persistent connections between HTTP clients and servers. The WebSocket capability in Robot Raconteur allows for standard web browsers to run a Robot Raconteur client and connect to Robot Raconteur services. While this is a useful capability, it also exposes the service to some risk of Cross-Site Scripting (XSS) attacks.

XSS attacks are when a webpage, typically an ad or some form of trap, downloads a JavaScript program that attacks another website. An example of an attack on a Robot Raconteur service is a webpage randomly attempting to connect on the local network and then trying to break into devices it finds. WebSockets protect against XSS by using an "Origin" policy. The "Origin" of the software attempting to connect is included in the WebSocket connection header by the web browser. The origin is based on web address of the webpage that is making the connection. Robot Raconteur by default only allows connections that are local files, extensions, or from the official Robot Raconteur website. Origins can also be added. More details on the origin control can be found in the "Introduction to Robot Raconteur using Python" manual.

Because of the risk of WebSocket XSS attacks, WebSocket connections can be disabled in situations where browser clients are not required. This can be done by setting `TcpTransport.AcceptWebSockets` to `False`.

6. Keep software up to date

Network security is a rapidly evolving technology and various vulnerabilities are frequently found that require patching. Please monitor threats to TLS and watch for emails from Robot Raconteur warning of patches that may be required.

5 Conclusion

This document serves as a basic introduction to the Robot Raconteur security features. More information can be found at http://robotraconteur.com. Network security is a rapidly evolving technology and various vulnerabilities are frequently found that require patching. Please monitor threats to TLS and watch for emails from Robot Raconteur warning of patches that may be required.